FH-OÖ Hagenberg/HSD
Software Entwurf 2, SS 2013
Übung 7

HSD FH-HAGENBERG

Name: Abgabetermin: KW 21

Mat.Nr: Punkte:

Übungsgruppe: korrigiert:

Aufwand in h:

Beispiel 1 (12 Punkte) Verwaltung von Wetterstationen: Implementieren Sie die Verwaltung von Wetterstationen aus Übung 4, Beispiel 2 unter Verwendung einer `std::list`, und zwar mit folgender Schnittstelle:

```
class WeatherStations {
public:
   // Constructors, Destructor, assignment operator (if necessary)
   //...

  // Inserts by name a new weather station.
  // If a weather station already exists, it is replaced.
  void Add(WeatherStation const& ws);

  // Removes a weather station
  bool Remove(WeatherStation const& ws);

  // Returns the number of weather stations
  size_t GetNrStations() const;

  // Prints all weather stations
  void PrintAll(std::ostream& out) const;

  // Print coldest/warmest weather station
  void PrintColdest(std::ostream& out) const;
  void PrintWarmest(std::ostream& out) const;

private:
  // member variables
  // ...
};
```

Überlegen Sie sich, ob Sie Default Constructor, Copy Constructor, Destructor und Assignoperator selbst implementieren müssen oder ob Sie mit den vom Compiler generierten Varianten auskommen.

Verwalten Sie die Wetterstationen in einer sortierten Liste, d.h. fügen Sie alle Wetterstationen aufsteigend nach dem Namen sortiert ein.

Achten Sie im Testtreiber darauf, dass alle Methoden ausführlich getestet werden.

Beispiel 2 (12 Punkte) STL-Algorithmen: Schreiben Sie ein C++ Programm, das zunächst mit Hilfe des Zufallszahlengenerators ganze Zahlen im Bereich zwischen 0 und 10 erzeugt und daraus Objekte der Klasse `Element` generiert und diese entsprechend der Erzeugungsreihenfolge in einem `std::vector` ablegt. Die Klasse `Element` speichert neben dem eingelesenen Wert auch die Erzeugungsreihenfolge (Index).

Kopieren Sie den Vektor in einen weiteren Vektor und sortieren Sie die Elemente beider Container nach dem Wert aufsteigend! Verwenden Sie für den ersten Vektor den Algorithmus `std::sort` und für den zweiten Vektor den Algorithmus `std::stable_sort`!

Die Ausgabe könnte für 40 erzeugte Zufallszahlen folgendermaßen aussehen:

```
unsorted values:
6-0 10-1 9-2 0-3 7-4 10-5 7-6 7-7 1-8 5-9 5-10 7-11 6-12 7-13 0-14
8-15 8-16 9-1 7 1-18 9-19 9-20 9-21 0-22 10-23 7-24 8-25 6-26 2-27
7-28 7-29 6-30 1-31 8-32 2-33 4-34 4-35 1-36 6-37 1-38 8-39

ascend stable sorted values:
0-3 0-14 0-22 1-8 1-18 1-31 1-36 1-38 2-27 2-33 4-34 4-35 5-9 5-10
6-0 6-12 6-26 6-30 6-37 7-4 7-6 7-7 7-11 7-13 7-24 7-28 7-29 8-15
8-16 8-25 8-32 8-39 9-2 9-1 7 9-19 9-20 9-21 10-1 10-5 10-23

ascend sorted values:
0-3 0-14 0-22 1-8 1-18 1-31 1-36 1-38 2-27 2-33 4-34 4-35 5-9 5-10
6-0 6-26 6-12 6-30 6-37 7-28 7-4 7-6 7-7 7-11 7-13 7-24 7-29 8-15
8-16 8-39 8-32 8-25 9-21 9-2 9-19 9-17 9-20 10-5 10-1 10-23
```

Die Ausgabe der Elemente des Containers soll mit dem `copy`-Algorithmus ermöglicht werden! Überschreiben Sie dazu den Ausgabeoperator entsprechend!

Kopieren Sie den Ausgangsvektor in eine `std::list` und sortieren sie die Elemente nach dem Wert aufsteigend mit der Sortierfunktion der Liste und geben Sie die Werte ebenfalls aus. Welche Aussage können Sie treffen?

Bestimmen Sie mit Hilfe von `std::equal_range` die Häufigkeit der Zufallszahlen im Container und geben Sie diese ebenfalls aus:

```
random number (0): 1
```

```
random number (1): 2
random number (2): 5
random number (3): 5
random number (4): 3
random number (5): 6
random number (6): 5
random number (7): 3
random number (8): 2
random number (9): 4
random number (10): 4
```

Allgemeine Hinweise: Legen Sie bei der Erstellung Ihrer Übung großen Wert auf eine **saubere Strukturierung** und auf eine **sorgfältige Ausarbeitung!** Verwenden Sie immer **Module**, um den Testtreiber und die eigentliche Implementierung zu trennen! Dokumentieren Sie alle Schnittstellen und versehen Sie Ihre Algorithmen an entscheidenden Stellen ausführlich mit **Kommentaren! Testen** Sie ihre Implementierungen ausführlich! Geben Sie **Lösungsideen** an!